SELF-SERVICE CATALOG MANAGER FOR STORES IMPLEMENTED ON A COMMUNICATIONS NETWORK

Inventors: Tim Roberts
Jyh-Herng Chow
Sara Hicks
Jimmy Duvall

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims priority under 35 U.S.C. § 119(e) to U.S. Provisional Patent Application Serial No. 60/504,577, "Self-Service Catalog Manager For Stores Implemented On A Communications Network," filed September 19, 2003. The subject matter of the foregoing is incorporated herein by reference in its entirety.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0002] This invention relates generally to catalog management for stores implemented on a communications network, for example as can be used to assist merchants to manage catalogs for Internet stores.

2. Description of the Related Art

[0003] The transfer of information over computer networks has become an increasingly important means by which institutions, corporations, and individuals do business. Computer networks have grown over the years from independent and isolated entities established to serve

the needs of a single group into vast internets that interconnect disparate physical networks and allow them to function as a coordinated system. Currently, the largest computer network in existence is the Internet. The Internet is a worldwide interconnection of computer networks that communicate using a common protocol. Millions of computers, from low-end personal computers to high-end super computers, are connected to the Internet.

[0004] The Internet has evolved to serve a variety of interests and forums. In particular, the Internet is rapidly transforming into a global electronic marketplace of goods and services as well as of ideas and information. This transformation of the Internet into a global marketplace was driven in large part by the introduction of an information system known as the World Wide Web ("the web"). The web is a distributed database designed to give wide access to a large universe of documents. The database records of the web are in the form of documents known as web pages. These web pages typically reside on web servers and are accessible via the Internet. Computers connected to the Internet may access the web pages via a program known as a web browser, which has a powerful, simple-to-learn graphical user interface. One powerful technique supported by the web browser is known as hyperlinking, which permits web page authors to create links to other web pages that users can then retrieve by using simple point-and-click commands on the web browser.

[0005] Web pages may be constructed in any of a variety of formatting conventions, such as Hyper Text Markup Language (HTML), and may include multimedia information content such as graphics, audio, and moving pictures. Any person with a computer and a connection to the Internet may access any publicly accessible web page. Thus, a presence on the World Wide Web (e.g., an Internet store) has the capability to introduce a worldwide base of consumers to businesses, individuals, institutions and other merchants seeking to advertise their products and services to potential customers. Furthermore, the ever increasing sophistication in the design of web pages and supporting infrastructure, made possible by the exponential increase in data

transmission rates and computer processing speeds, makes the web an increasingly attractive medium for these stores.

[0006] The availability of powerful new tools that facilitate the development and distribution of Internet content has led to a proliferation of information, products, and services offered on the Internet and dramatic growth in the number of consumers and merchants using the Internet. Commerce conducted over the Internet has grown and is expected to continue to grow dramatically. As a result, the Internet has emerged as an attractive new medium for merchants offering various products and services to reach these large numbers of consumers.

[0007] In particular, small businesses, especially those that address highly targeted niche markets, may benefit substantially from a store presence on the Internet (or other similar computer network). The cost of a store and advertising on the Internet can be low compared to other alternatives, and merchants potentially can reach a very wide audience (or a highly targeted audience). However, existing technology for supporting stores, especially administrative functions such as catalog management, are not well suited for smaller merchants. Much of the existing technology is targeted for managing catalogs and inventory for very large merchants. This technology tends to be too expensive for smaller merchants to afford and unnecessarily complex for them to use. In particular, it is often desirable for smaller merchants to have self-serve capability to manage their catalogs and perform other administrative functions. Currently available technology is not well suited for this application.

[0008] Thus, there is a need for new approaches to catalog management and other administrative functions in order to better serve these merchants.

SUMMARY OF THE INVENTION

[0009] The present invention overcomes the limitations of the prior art by providing a computerized self-service platform that assists merchants in creating and managing stores on a computer network, such as the Internet. The self-service aspect is beneficial because it reduces the cost of administering stores. It also increases operational flexibility. In one implementation, changes to the store can be made at any time and in real-time or near real-time, thus allowing the merchant to respond more rapidly to changing conditions.

[0010] In one aspect of the invention, a system for managing a store includes a product database and an ecommerce storefront. The product database stores information concerning items (including services and other non-product offerings) offered for sale through the store. The ecommerce storefront stores information concerning the presentation of the store to customers (i.e., the storefront). Some of the storefront information references information stored in the product database. A product administration module allows the merchant (or other users) to manage the product database on a self-serve basis. Separating the storefront from the product database has numerous advantages. For example, each part can now be handled by different people (e.g., a database administrator for the product database and a web designer for the storefront). The separation also makes it easier to reuse components. For example, some or all of a product database that is built for one store can potentially be reused with a different storefront having an entirely different look and feel.

[0011] In one specific implementation, the store is implemented on the Internet. The product database is divided into two parts: a product catalog and inventory records. The user can administer the product catalog and inventory records via a self-serve catalog manager that is accessible through a URL. The storefront includes web pages, which are served to customers by an ecommerce server. The web pages can contain store tags, which reference information from the product catalog and/or inventory records. A store tag server is used to resolve the store tags.

[0012] In another aspect of the invention, the product database stores its information in the form of tables. Each table is organized into items and items may have options. In one specific data structure, each row in a table corresponds to one item and options for an item are encoded and stored as a single entry in the row. However, inventory preferably is tracked separately for each option. For example, if one item is a certain type of shirt and the options are different sizes, a single web page can be built with respect to the item but inventory is tracked separately for each shirt size.

[0013] Other aspects of the invention include subsystems, data structures and methods related to the systems described above.

BRIEF DESCRIPTION OF THE DRAWING

[0014] The invention has other advantages and features which will be more readily apparent from the following detailed description of the invention and the appended claims, when taken in conjunction with the accompanying drawing, in which:

[0015] FIG. 1 is a block diagram of an example system suitable for use with the present invention.

[0016] FIG. 2 is a block diagram of an example implementation of a store of FIG. 1.

[0017] FIG. 3 is a graphical depiction of an initial web page for a self-service platform for catalog management.

[0018] FIGS. 4A-4E are graphical depictions of web pages for viewing items in a catalog.

[0019] FIGS. 5A-5B are graphical depictions of web pages for adding an item to a catalog.

[0020] FIGS. 6A-6D are graphical depictions of web pages for editing an item in a catalog.

[0021] FIG. 7 is a graphical depiction of a web page for viewing tables in a catalog.

[0022] FIGS. 8A-8F are graphical depictions of web pages for adding a table to a catalog.

[0023] FIGS. 9A-9B are graphical depictions of web pages for editing a table in a catalog.

[0024] FIGS. 10A-10D are graphical depictions of web pages for importing data to a catalog.

[0025] FIG. 11 is a graphical depiction of a web page for exporting data from a catalog.

[0026] FIGS. 12A-12B are graphical depictions of web pages for managing inventory of items in a catalog.

[0027] FIG. 13 is a graphical depiction of a web page verifying publication of a catalog.

[0028] FIG. 14 is a block diagram of an example store according to the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0029] FIG. 1 is a block diagram of an example system 100 suitable for use with the present invention. Generally speaking, the system 100 includes a number of sites 110A-N, customers 130A-N and merchants 140A-N that communicate with each other over a network 120. The merchants 140 maintain stores 150A-N on the sites 110. The customers 130 shop at the stores 150 via their storefronts on the sites 110. The merchants 140 access the stores 150 (e.g., to perform administrative functions) via a self-service platform, which can be implemented as part of a site 110.

[0030] In one specific embodiment, the network 120 is the Internet. The sites 110 include web sites, for example Yahoo! Store. The customers 130 include individuals who access the Internet, typically by web browsers 135 such as Netscape's Navigator or Microsoft's Internet

Explorer. The merchants 140 maintain Internet stores 150, which are hosted on the sites 110. The host site may be maintained by the merchant itself or by a third party (e.g., Yahoo!). Each site 110 may contain multiple stores 150. The merchants 140 typically access the self-service platform (implemented on sites 110) by web browsers 135. In some cases, the customers 130 and merchants 140 can access the sites 110 using other means, for example by software agents, programmatic interfaces or communications channels outside of the network 120. The customers 130 shop at the stores 150 via the Internet, primarily by requesting web pages from the sites 110 and/or submitting forms to the site 110.

[0031] FIG. 1 is simplified for clarity. For example, the sites 110, customers 130 and merchants 140 are shown as separate entities. In fact, the same entity may play one or more roles. Entities may also take on different roles in different contexts. In addition, the different roles can be distributed and/or divided among many different entities. The site 110 itself may also be distributed for redundancy and/or performance reasons. For example, large sites such as Yahoo! sites typically run different web properties from different servers and may use for example multiple servers, databases, load balancers, etc.

[0032] As further clarification, although the Internet will be used as the primary example in this disclosure, the invention may be used with other systems also, for example some POS systems or in store retail systems. For example, the entities 110, 130 and 140 may communicate with each other over separate communications networks or dedicated communications channels, rather than (or in addition to) the common network 120 of FIG. 1. Alternately, various parts of system 100 may be implemented by mobile components and may not be permanently attached to a communications network. For example, entities may interact with each other via a wireless connection. As a final example, the pages can be based on protocols other than the web.

[0033] FIG. 2 is a block diagram of an example implementation of a store, based on the Internet scenario described above. The storefront for the store is the presentation of the store to the customer. Other parts of the store include administrative functions (such as maintaining a

catalog of items for sale in the store, inventory available for each item, and pricing) and back-end processing (such as credit card processing and generation of invoices, as applicable).

[0034] In FIG. 2, information about the store is stored in two databases 210 and 260. The product database 210 ("product" is meant to also include services and other non-product offerings) includes information about the store catalog, such as items available for sale, SKU number, different options for each SKU, inventory available, pricing, etc. In one approach, the product database is organized as tables, each table having fields that can vary from table to table. Each table contains items, with items typically defined by SKU number. Differences within a single SKU number typically appear as options within one item. For example, one table might be Apparel. Items in the Apparel table might include different types of sweaters, shirts, pants, etc. The fields in this table might include SKU, designer label, price, etc. Options for apparel might include size and color, with inventory listed separately for each option. Another table might be Books, with each item being a different title. Fields in the Books table might include ISBN number, hardback vs. paperback, author, publisher, etc.

[0035] Using multiple tables makes it easier to define different fields for each table. For example, if only a single table were used, the ISBN number field would be defined for all items, even though it is a meaningless field for Apparel items. It would be wasted space in the product database 210. The use of options to further subdivide an item also results in efficiency. For example, if one type of sweater is to be placed on sale, this can be achieved by editing the catalog with respect to the item in its entirety, rather than individually editing the catalog entries for each size, color, etc. Furthermore, there can be a single presentation of the item in the storefront (e.g., one image is used to show a certain style of sweater, rather than having different images for every size and color), although inventory and/or pricing are maintained separately for each option. Other groupings can also be supported. For example, groups can be defined as including certain items and/or options (or other groups). Manipulating the group then affects the individual members in the group. For example, the group Sharks Apparel can be defined to include Apparel

items with the San Jose Sharks branding. When the Sharks season ends, the group in its entirety can be placed on sale.

[0036] The ecommerce storefront 260 includes information about the storefront, such as the actual pages to be served to shopping customers. In one embodiment, the web pages access information in the product database 210 by using store tags, as will be further described below. For example, on the web page displaying a book, rather than hardwiring "$25.99" in the HTML for the web page, the store tag that points to the book price in the product database 210 can be used instead. Thus, if the price is changed in the product database 210, it is automatically updated on the web page also. One advantage of using store tags (dynamic elements in a static page) to accomplish this task, as opposed to a web page that is dynamically generated, is that it has a static URL. Many web crawlers cannot index pages that are generated in response to queries since the web crawler itself does not generate queries. Thus, using store tags allows the page to be indexed.

[0037] Separating the storefront from the product catalog has numerous advantages. For example, managing the product catalog and building the storefront requires two different skill sets. By separating these tasks, each can be handled by specialists with the appropriate skill sets and using specialized tools. The separation can also lead to the reuse of components. For example, some or all of a product catalog that is built for one store can potentially be reused with a different storefront having an entirely different look and feel. To take this one step further, many stores will have common elements, either with respect to their catalogs or with respect to the functionality of their storefronts. Templates can be developed to exploit these commonalities, thus allowing merchants to quickly build their stores. Additionally, this allows for catalog data elements to be referenced from not just an associated storefront in the same hosting environment, but also from other sources such as a storefront in a different hosting environments (e.g., if the web servers have been modified to access the catalog data via store

tags) and aggregated shopping environments where customers can browse products from different merchants.

[0038] Returning to FIG. 2, customers 130 access the store via the ecommerce server 270, which serves the appropriate web pages. The ecommerce server 270 can be a web hosting server. The ecommerce server 270 retrieves the desired web pages from the database 260 and can also access the product database 210 as needed, for example to resolve store tags in the HTML.

[0039] Merchants 140 can access the store via the product administration module 230 and the storefront editor 220. The storefront editor 220 is a tool used to manage the presentation of the store site. For example, changes to the look and feel of the storefront (i.e., the web pages making up the storefront) can be made via the storefront editor 220. The editor 220 can access the product database 210 as needed. In some implementations, this tool can perform the combined actions of managing presentation of the product data and the data itself. The product administration module 230 is a tool used to manage the product database. Changes to the items listed in the catalog, pricing, etc. can be made via the administration module 230.

[0040] The administration module 230 preferably is self-serve, meaning that the merchant can access the administration module 230 himself and make changes needed to the product database 210. This is especially beneficial for smaller merchants. It often is not cost effective for personnel to directly support the administrative activities of smaller merchants. Yet, smaller merchants must have the ability to maintain their catalogs. The self-service aspect reduces the administrative cost by allowing smaller merchants to help themselves. The self-service aspect also allows the merchant to more easily and more immediately change his catalog. If the overall system is real-time or near real-time, as is the case in this example, any changes made by the merchant will be reflected immediately or near-immediately in his storefront.

[0041] In one implementation, the components shown in FIG. 2 are not implemented separately for each store. Rather, multiple merchants share many common components. For

example, each merchant 140 does not have a dedicated storefront editor 220, administration module 230, or ecommerce server 270. These components are used to support all stores hosted by a site. Each merchant 140 also does not have a dedicated database 210 and 260. Rather, common databases are used but each merchant 140 has a dedicated section within the common databases.

[0042] FIGS. 3-13 are graphical depictions of web pages that illustrate an example user interface for the product administration module 230. This self-service platform allows users 140 to manage their catalogs. This particular example is in the context of catalogs for stores hosted by Yahoo! Store, but the invention is not limited to this specific example. In the example of FIGS. 3-13, the self-service platform is referred to as Catalog Manager, which is located on the Yahoo! Store site. In an alternate implementation, the self-service platform is located on a site run by a different entity (e.g., a service provider to Yahoo!). Alternately, the self-service platform can manage catalogs and/or perform administrative functions for stores maintained on many different sites.

[0043] FIG. 3 shows a login page for Catalog Manager. From this page, the user has a number of options. He can get more information about his account via link 310. In area 320, he can access other resources, including the store tag hub (described below). The catalog management functions 330 in this example are divided into five main areas: managing items, managing tables, importing/exporting data, inventory functions, and publication. On later web pages, these five areas can be accessed by tabs on the web page. FIGS. 4-6, 7-9, 10-11, 12 and 13 illustrate each of these five functional areas in more detail.

[0044] FIGS. 4A-4E show pages for viewing items in the catalog. Different subsets of items can be selected for viewing. FIG. 4A shows a web page where all tables 410 are selected for viewing and no other limits are put on the set of items to be viewed. As a result, all items in the catalog are part of the view set and, in this example, they are displayed in alphanumeric order. In FIG. 4B, the view set is restricted to the Cookies table 420 (note that the table

assignments for the items are not consistent between FIGS. 4A and 4B). In FIG. 4B, there is no need to display a separate Tables field since all items in this view set are from the same table. In FIG. 4B, the item alphabetcookie has three options, which are currently displayed. The display can be expanded or collapsed to view or hide the different options.

[0045] The view set can also be defined in other ways. For example, the Search button 414 can be used to search for specific items or specific classes of items. FIG. 4C shows a form used to search for keywords within a particular table and field. FIG. 4D is an example of the display resulting from a search for the term "cookie" in the field "Name" for all tables. Other types of searches can also be performed. For example, the view set could be defined by all items that are currently on sale, or all taxable items, or all items in the Table "Cookies" that also have the word "Chocolate" in the name. Standard search techniques can be used to define and fill search requests.

[0046] The layout of the display itself can also be changed by the user. For example, the fields displayed in FIG. 4A are ID, Name, . . . Table. These fields can be customized. FIG. 4E shows a user interface that allows the user to determine which fields should be displayed. The displayed fields can be different for different tables. Standard techniques can be used to modify the layout with respect to font, color, alignment, etc. The display of FIGS. 4A-4B is in the form of a table with 25 items shown per display page. The number of items per page can be varied. In other implementations, layouts other than table based can be used.

[0047] Returning to FIG. 4A, from the item display, the items can be edited. Clicking on a specific item allows the user to edit the catalog entries for that item. The "Edit These Items" button 416 edits selected items. Items can be selected by checking specific check boxes 412. Clicking "Check All" 415 selects all items. A specific "Change Prices" button 417 is available as a shortcut since changing price is a common edit. Items can be selected for editing in other ways, for example by highlighting selected items. The "Add Item" button 419 allows the user to add new items.

[0048] FIGS. 5A-5B show pages for adding an item. In this example, the "Add Item" button 419 was clicked while viewing items for the table TableX. Therefore, this new item will be added to TableX. FIG. 5A shows an entry form with the fields for TableX. Some of these fields are limited to menu choices (e.g., Taxable field is either Yes or No). Some can accept free form text (e.g., the Options field). Some can accept uploads (e.g., the Image and Icon fields). In the Options field, clicking on "Enter Individual Item Codes" leads to the entry form shown in FIG. 5B. This page allows the user to specify the different options available for this item. Once the form in FIG. 5A is completed, the user clicks "Save" or "Save and Add Another." The submitted information is checked, for example to ensure that all mandatory fields are completed. Once there are no errors, the new item can be added to the catalog. The user typically receives a confirmation.

[0049] Items can also be added in other ways. For example, data about items can be uploaded to the catalog rather than individually adding each item as shown in FIGS. 5A-5B. Alternately, templates can be built for different classes of items so that certain fields are already filled in, or scripts can be used to automate filling in fields.

[0050] FIGS. 6A-6D are pages used to edit items in a catalog. Referring to FIG. 4A, if the user selects a single item to be edited, this can be achieved by displaying the page shown in FIG. 5A but with the fields filled in with their current values. The user can then edit the fields. If the user selects multiple items to be edited, for example an entire table, a set selected by a search, or the entire catalog, the page shown in FIG. 6A can be more appropriate. In this case, the items and fields are shown in table format but the fields can be edited.

[0051] FIG. 6B shows a "verification" page when the user has requested to delete certain items. In this example, the user has selected four items from the display of FIG. 4A, and then clicked the "Delete" button 421. The page in FIG. 6B verifies the request, also noting that the item "crackers" has four options that will also be deleted. This feature can be especially useful if the user has defined groups of items and then requests to delete the entire group. The verification

can explicitly list all the items in the group to confirm that the user indeed desires to delete all of the items. Returning to FIG. 6B, if the user does not want to delete all four items, he can uncheck the items that are not to be deleted.

[0052] FIG. 6C shows a form when the user has requested to move certain items from one table to another. The user has selected four items from the display of FIG. 4A, and then clicked the "Move" button 423. The user specifies the destination table using the form of FIG. 6C. In some cases, a move can result in the deletion of fields, for example if the destination table does not include a field contained in the same table. In these cases, a verification page can alert the user to this situation before moving the items.

[0053] FIG. 6D shows a form for certain types of price changes. The "Change Prices" button 417 in FIG. 4A activates this form. Using this page, the user can apply a certain percentage discount to all tables or just to certain tables. The user can also turn sale pricing on and off. Pricing can also be edited using the techniques described above for other fields.

[0054] Turning now to table management, FIG. 7 shows a page for viewing tables in the catalog. The layout, organization and manipulation of this web page (and other web pages for manipulating tables) are similar to the approach used to manipulate items, as shown in FIGS. 4-6. This is beneficial since an intuitive and simple user interface is especially desirable for a self-service platform. Users ideally will be able to learn the user interface on their own and with limited documentation. This example displays all tables in the catalog, along with their name, description and the number of items in the table. The check boxes serve the same function as in FIGS. 4, allowing the user to select tables.

[0055] Many of the options described with respect to viewing items are also applicable to viewing tables. For example, the user may want to select a subset of tables for viewing, to expand or collapse parts of the table display, to perform searches, and/or to change the layout of

the table display. In this example, fewer options are provided for manipulating the table display than are provided for the item display, mainly because far fewer tables are expected.

[0056] The "Create New Table" button 719 allows the user to add new tables. FIGS. 8A-8F show pages for adding a table. In FIG. 8A, the user enters the table name and a description of the table.

[0057] Next, the user defines which fields are included in the table. In this example, the fields are divided into subsets: mandatory fields, store fields, shopping fields and custom fields. FIGS. 8B-8E show entry forms for each respective subset. In this example, each field is defined by its name, its format (number, text, yes-no, etc.), whether it is required to be filled in for each item (which is different than whether the field is required to be included as part of a table), and a default value, if any. FIG. 8B shows the entry form for mandatory fields in this example. Mandatory fields must be included for all tables. The name and format are already defined. Most mandatory fields are also required for all items.

[0058] FIG. 8C shows the entry form for store fields. In this example, store fields are predefined in order to standardize fields across users. However, these fields are not required to be included with all tables. Thus, this form also includes a column "Include" which determines whether a field is to be included in this particular table. Shopping fields are handled similarly to store fields. FIG. 8D shows the entry form for shopping fields. Finally, FIG. 8E shows an entry form that allows the user to define custom fields.

[0059] FIG. 8F is a verification page for creating a table. The submitted information is typically checked as part of this process. The Edit buttons 810 permit the user to revise any information. Like items, tables can also be added in other ways.

[0060] In FIG. 7, clicking on a specific table allows the user to edit the information for that table. FIG. 9A is a web page used to edit a table in a catalog. This page operates similarly to the one in FIG. 8F. Clicking an Edit button brings up the corresponding section of the table, which

the user can then edit. When edits are complete, the user clicks the Save button. If the user is deleting fields, a verification page first reminds the user that information in the deleted fields will be lost, as shown in FIG. 9B.

[0061] FIGS. 4-9 were form-based approaches to editing a catalog. The catalog can also be modified in other ways. FIG. 10 shows an example where data is imported into a catalog. In this example, the import process is divided into two steps: upload and commit. The upload step includes converting the source data from its original form into the form compatible with the product database, for example defining which fields in the source data correspond to which fields in the catalog. The commit step includes combining the uploaded data with the existing database. In this example, there are two forms of commit: add, where new records are added to the product database and rebuild, where existing records are revised or replaced with the imported records.

[0062] FIG. 10A shows a web page that allows a user to upload data by clicking the "Upload" button. This brings the user to the form in FIG. 10B. Here, the user specifies the destination table within the catalog, whether the table will be added to or rebuilt, and the source file for the upload. In later pages, the user can also specify how fields in the source file map to those in the destination table. Clicking "Upload" initiates the upload (after verification). FIG. 10C is the results page where the upload encountered errors, which are shown in the "Warning and Errors" section. Separating the upload and commit processes allows the user to correct any warnings and errors before affecting the product database.

[0063] When the user is satisfied that the source data is uploaded correctly, he clicks "Commit," which then implements the add or rebuild, as appropriate. The commit step typically is verified before it executes. The verification page may include warnings regarding potential errors or information that will be lost. FIG. 10D shows a web page after the commit process has been performed. In this example, the user has committed the upload from FIG. 10C (i.e., the one

with the errors). As a result, the commit process also reflects these errors. It may be that the error warnings are erroneous.

[0064] Data can also be exported from the product database. FIG. 11 shows a form for downloading a table from a catalog.

[0065] Turning now to inventory management, FIGS. 12A-12B show web pages for managing inventory. Inventory management can be handled within the general framework for catalog management, for example as described above in FIGS. 3-11. However, in this example, inventory management also has dedicated web pages due to the importance of this function. In some implementations, inventory data is also stored in a database separate from the rest of the catalog data. That is, the product database may contain a separate inventory database.

[0066] FIG. 12A shows a web page for viewing inventory. The layout, organization and manipulation of this web page (and other web pages for editing inventory) is similar to the pages used to manipulate items and tables, as shown in FIGS. 4 and 7. This is beneficial since it makes the self-service platform easier to learn and use. This example displays inventory for all items in the table "Cookies." Clicking the "Edit" button 1216 allows the user to edit the inventory, for example as shown in FIG. 12B. The "Upload" and "Download" buttons 1222 and 1224 facilitate the import and export of inventory data.

[0067] FIGS. 3-12 show different web pages for assisting a merchant to manage his catalog. Typically, there will be two versions of the catalog. The published catalog is the version that is currently implemented on the site. It is the one that customers will experience when they visit the merchant's storefront on the site. The working catalog is a version that the merchant can use to make edits, revisions, etc. without affecting the customer's experience. In a typical architecture, the catalog management functions described above affect the working catalog, not the published catalog. When the merchant desires for the working catalog to go live,

he publishes the working catalog. FIG. 13 shows a verification page for publication of a working catalog.

[0068] FIGS. 3-13 illustrate one example of a self-service platform for catalog management. In this example, the pages generally adopt a common approach to viewing and editing the catalog. In addition, the user is not given unlimited freedom to manipulate the catalog. Rather, the platform is designed to automate the most common functions and to limit user access to sophisticated or complex manipulations. This is advantageous for a self-service platform that is intended to be used by large numbers of smaller merchants. The common approach to layout and editing makes the platform easy to learn and intuitive to use, even without training classes or lengthy documentation. The limited freedom for users reduces the risk that unsophisticated users will accidentally harm their own catalogs or that malicious users will intentionally do the same to the catalogs of others. It also reduces the amount of technical support required for the platform, thus reducing the cost to the end user.

[0069] In addition, the invention is not limited to the specific example of FIGS. 3-13. Many other variations will be apparent. For example, a different look and feel for the user interface can be used. The displays need not use a table format. They can contain images, audio and other types of information. They can be voice-activated or support other types of I/O. In addition, the user interface can be customized for each individual user. For example, users could customize the first page (or other pages) of Catalog Manager to suit their needs.

[0070] The interface for manipulating the catalog also need not be limited to web pages, forms and the import/export of data. Software automation can be supported. For example, the catalog can be designed to be manipulated via an API, a command set and/or scripts. These options make it easier for the catalog to be directly interfaced to other systems, for example the inventory management system at the merchant's warehouse or a master price sheet. Templates can also be provided to automate tasks that are likely to be common among many users.

[0071] In addition, the management functions implemented are not limited to those shown in FIGS. 3-13. For example, in alternate versions, pricing can be based on a more sophisticated schedule or table (e.g., pricing automatically changes over time, or with inventory remaining, or with quantity purchased). As another example, availability of items may automatically vary over time. The platform can also allow the user to set up alerts or notifications. For example, the user may desire an alert when inventory drops below a certain level (or remains above a certain level for too long). Different reports can also be generated for the user.

[0072] The catalog is also not limited to an organization based on tables, items and options. Other structures can be supported. For example, the user can define groups, or even groups of groups, etc. Alternately, an object-based structure can be used, where child objects inherit attributes from their parent objects.

[0073] Turning now to FIG. 14, FIG. 14 is a simplified block diagram of a store according to the invention. In this example (as is the case throughout this disclosure), the same component may be referred to as a module, server or other descriptive term, for example "hosting," "hosting module," "hosting server," or "hosting application." It should be understood that these components are not meant to be limited to a specific physical form. In most cases, the preferred implementation currently is software. However, depending on the specific application, they can be implemented as hardware, firmware, software, and/or combinations of these. Furthermore, different components can share common sub-components or even be implemented by the same sub-components. There may or may not be a clear boundary between different components.

[0074] Comparing FIG. 14 with FIG. 2, the product database 210 is implemented as a product catalog 812 and inventory records 814. The product catalog 812 includes information for the store's catalog, for example items offered for sale, images of the items, pricing, text description, different options, etc. The product catalog 812 uses the structure of tables, items and options as described above. The information in the product catalog 812 is stored as tables; each row in a table corresponds to one item. Relational databases are not used in this example.

Options are handled by encoding the options to produce a single string and then storing that string in the Options field of the table. The string is decoded when information about options is desired. The inventory records 814 are separated from the rest of the catalog and are stored using conventional techniques.

[0075] The ecommerce storefront 260 stores the web pages for the presentation to the customer. The web pages typically contain references to other locations, for example images, dynamic feeds, advertising and/or store tags. Store tags are references to information from the product database 210. The storefront editor 220 assists the merchant to build and maintain his ecommerce site. It can include templates to automate the building of the merchant's storefront.

[0076] The product administration module 230 includes modules that assist the merchant to manage the product database. The merchant accesses the administration module 230 via a URL. The specific modules shown in FIG. 14 correspond to the functions described in FIGS. 3-13. The item management module 836, table management module 835 and import/export module 837 implement the item management, table management and import/export functions described in FIGS. 4-6, 7-9 and 10-11, respectively. These modules primarily interface with the product catalog 812 in order to edit the catalog. The inventory management module 839 implements the inventory functions described in FIG. 12. It primarily interfaces with the inventory records 814.

[0077] The publication module 832 is responsible for publication of the catalog, as described in FIG. 13. The published catalog is depicted as box 833 in FIG. 14. Other components access the published catalog 833 via the publication module 832. For example, the ecommerce server 270 interacts with the publication module 832 to access the published catalog 833, rather than accessing the product catalog 812 directly. The box 833 in FIG. 14 is merely a graphical depiction. It is not meant to imply that the published catalog 833 is stored as part of the publication module 832 or even that the published catalog 833 is separate from the product catalog 812. For example, in one implementation, a common database is used to store both the

published catalog and the "working" catalog. Each row in each table is flagged as either published or not.

[0078] If the merchant desires, he can export the published catalog 833. Thus, for example, the merchant can export the published catalog 833 and then use development tools other than the storefront editor 220 to build the store site. Examples of other development tools include Macromedia's DreamWeaver, Microsoft's FrontPage and other HTML editors. When the web pages are loaded back into the system, store tags provide the linkage to information contained in the product database. The published catalog 833 can also be exported to other marketplaces. Separating the product catalog from the storefront look and feel allows the merchant to build the catalog once and reuse it with different storefronts in many marketplaces.

[0079] In FIG. 14, the store tag hub 831 is a module that assists the merchant in the use of store tags. The store tag hub 831 can contain information describing the concept of store tags, a tutorial on their use and/or tools to facilitate their use. For example, the store tag hub 831 can include wizards to help the merchant identify the correct store tag for a particular piece of information.

[0080] On the customer side, the ecommerce server270 includes a hosting server 874 and a store tag server 872, as well as other servers. An order server 876 and a billing server 877 are shown in FIG. 14, although other components (e.g., image server, search server, library server, redirect server, wallet technology, credit card processing, tax and shipping and handling tables, etc.) typically will also be used. The ecommerce server 270 accesses the various data sources. In this example, the ecommerce server 270 can access the inventory records 814 and ecommerce storefront 260, and can obtain the published catalog 833 via the publication module 832.

[0081] The hosting server 874 is responsible for serving the storefront web pages to the customer. It retrieves the web pages from the ecommerce storefront 260. Store tags in the retrieved web pages are passed to the store tag server 872, which resolves the store tags based on

the published catalog 833. In this implementation, the store tags are server-side include (SSI) HTML tags. The store tag server 872 is a web server that dynamically serves data elements from the product catalog in the HTML page. This is done when the page is served but while keeping a static page URL. The hosting server 874 communicates with the store tag server 872 via whproxy.

[0082] To reduce unnecessary communication, the results of store tag expansion are cached. In one approach, when the hosting server 874 first encounters a store tag for a particular store on a web page, it checks for a cached copy of the page (e.g., saved as a hidden file with the same name and location as the original file). The cached copy is used if it has a more recent time stamp than the retrieved web page and the catalog has not been updated in the interim. In this case, no further requests need be sent to the store tag server 872 for this particular store and page. If the cached page is out-of-date or there is no cached page, the current page is also stored as a cached copy. Store tags are expanded in the cached copy. In an alternate approach, store tags can be cached on a tag-by-tag basis rather than on a page-by-page basis.

[0083] Although the detailed description contains many specifics, these should not be construed as limiting the scope of the invention but merely as illustrating different examples and aspects of the invention. It should be appreciated that the scope of the invention includes other embodiments not discussed in detail above. Various other modifications, changes and variations which will be apparent to those skilled in the art may be made in the arrangement, operation and details of the method and apparatus of the present invention disclosed herein without departing from the spirit and scope of the invention as defined in the appended claims. Therefore, the scope of the invention should be determined by the appended claims and their legal equivalents. Furthermore, no element, component or method step is intended to be dedicated to the public regardless of whether the element, component or method step is explicitly recited in the claims.